Một ví dụ về xây dựng mô hình dựa trên dữ liệu batdongsan_com_vn

August 3, 2022

0.1 Lấy dữ liệu đã được xử lí từ file có sẵn

[ ]:
```
import pandas as pd
data = pd.read_csv('newdfBDS.csv')
print(len(data))
data.head(2)
```

```
81162
```

[ ]:
```
         id  month                    project  …     district      ward
price
0  28086120     12  Vinhomes Smart City Đại Mỗ  …  nam tu liem    dai mo
1560.0
1  28088954     12               Goldmark City  …  bac tu liem  phu dien
3300.0

[2 rows x 12 columns]
```

0.2 Lọc để chỉ làm việc với dữ liệu tháng 8 - 12/2020

Thời điểm xây dựng mô hình là tháng 1/2021 nên tác giả chỉ lấy dữ liệu trong khoảng thời gian gần đó. Bạn đọc có thể xem xét dữ liệu thuộc các khoảng thời gian khác để đánh giá thêm

[ ]:
```
data_aug = data.loc[data['month'] == 8]
data_sep = data.loc[data['month'] == 9]
data_oct = data.loc[data['month'] == 10]
data_nov = data.loc[data['month'] == 11]
data_dec = data.loc[data['month'] == 12]
data_new = pd.concat([data_aug, data_sep, data_oct, data_nov, data_dec])
print('So luong ban ghi thang 8 - 12: ', len(data_new))
data_new.head(3)
```

```
So luong ban ghi thang 8 - 12:  56969
```

[ ]:

	id	month	project	…	district	ward	price
9434	26489651	8	Hope Residence	…	long bien	phuc dong	1200.0
9435	26602206	8	NaN	…	nam tu liem	my dinh 2	810.0
9436	26459338	8	Khu đô thị mới Xa La	…	ha dong	phuc la	1700.0

[3 rows x 12 columns]

Một vài bước tiền xử lí dữ liệu trước khi huấn luyện mô hình

[ ]:
```
data_new.describe()
```

[ ]:

	id	month	…	bathrooms	price
count	5.696900e+04	56969.000000	…	56969.000000	56969.000000
mean	2.680693e+07	9.835156	…	1.898313	2430.901294
std	1.569313e+06	1.365532	…	0.416280	1050.929709
min	6.449162e+06	8.000000	…	1.000000	350.000000
25%	2.662292e+07	9.000000	…	2.000000	1550.000000
50%	2.709917e+07	10.000000	…	2.000000	2325.000000
75%	2.754646e+07	11.000000	…	2.000000	3200.000000
max	2.812856e+07	12.000000	…	4.000000	5000.000000

[8 rows x 6 columns]

[ ]:
```
#Chuyen gia tri truong 'month' thanh dang string (dạng categorical)
data_new = data_new.astype({"month": str})
```

[ ]:
```
data_new.head()
```

[ ]:

	id	month	…	ward	price
9434	26489651	8	…	phuc dong	1200.0
9435	26602206	8	…	my dinh 2	810.0
9436	26459338	8	…	phuc la	1700.0
9437	26608859	8	…	duong xa	1870.0
9438	26790642	8	…	NaN	420.0

[5 rows x 12 columns]

[ ]:
```
#Xử lí các trường dạng số - Chuẩn hóa giá trị
from sklearn.preprocessing import MinMaxScaler
mmscaler = MinMaxScaler()
X_num = data[['square', 'bedrooms', 'bathrooms']]
mmscaler.fit(X_num)

X_num = mmscaler.transform(X_num)
```

[ ]:
```
##Lọc lấy các trường category để xử lí :">> (Đưa về các biến giả dummy)
```

```
X_cat = data_new.drop(['square', 'bedrooms', 'bathrooms', 'price', 'id'],␣
↪axis=1)
X_cat = X_cat[['project', 'investor', 'district', 'ward']]

import pandas as pd
X_cat_new = pd.get_dummies(data = X_cat)
X_cat_new.head()
```

[ ]:

	project_6th Element	…	ward_yet kieu
9434	0	…	0
9435	0	…	0
9436	0	…	0
9437	0	…	0
9438	0	…	0

[5 rows x 1050 columns]

[ ]:
```
#X là ma trận chứa thông tin các trường thuộc tính
#y là vecto giá tương ứng

import numpy as np
X = np.concatenate([X_num, X_cat_new], axis=1)
y = data_new[['price']].to_numpy()
```

[ ]:
```
#Lưu lại X, y đã xử lý để phòng sự số xảy ra
import pickle as pkl
pkl.dump(X, open('X.pkl', 'wb'))
pkl.dump(y, open('y.pkl', 'wb'))
```

[ ]:
```
#Khi cần, load lại X, y từ file, tương ứng là ma trận thuộc tính và vecto nhãn␣
↪của dữ liệu.

import pickle as pkl
X = pkl.load(open('X.pkl', 'rb'))
y = pkl.load(open('y.pkl', 'rb')).ravel()
```

[ ]:
```
print(X.shape)
print(y.shape)
```

```
(56969, 1053)
(56969,)
```

[ ]:
```
#Chia train/test
x_train , x_test , y_train , y_test = train_test_split(X , y , test_size = 0.
↪5,random_state =2, shuffle = True)
```

0.3 Xây dựng mô hình

0.3.1 Ham danh gia mo hinh tren du lieu tap train va tap test

[ ]:
```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error

def evaluate(model, x_train, x_test, y_train, y_test):

  y_pred =  model.predict(x_test)
  r2 = r2_score(y_test, y_pred) ### Tap test
  mse = mean_squared_error(y_test, y_pred)
  mae =  mean_absolute_error(y_test,  y_pred)
  print('r2: ', r2, '\nmse: ', mse, '\nmae: ', mae)
  print('----------------------------------------')

  y_train_pred = model.predict(x_train) ### Tap train
  r2 = r2_score(y_train, y_train_pred)
  mse = mean_squared_error(y_train, y_train_pred)
  mae =  mean_absolute_error(y_train, y_train_pred)
  print('r2: ', r2, '\nmse: ', mse, '\nmae: ', mae)

  print(type(y_train))
  print(type(y_train_pred))
```

0.3.2 Khởi tạo mô hình và huấn luyện:

[ ]:
```
# Mô hình với các siêu tham số khởi đầu
from sklearn import ensemble
clf = ensemble.GradientBoostingRegressor(n_estimators = 400, max_depth = 5,␣
 ↪min_samples_split = 2,
         learning_rate = 0.1, loss = 'ls')
```

[ ]:
```
clf.fit(x_train, y_train)
```

[ ]:
```
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.1, loss='ls', max_depth=5,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=400,
                          n_iter_no_change=None, presort='deprecated',
                          random_state=None, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
```

[ ]:
```
evaluate(clf, x_train, x_test, y_train, y_test)
```

```
r2:  0.9309047046986271
mse:  76549.18758059527
mae:  197.11983744565003
----------------------------------------
r2:  0.9365339022679423
mse:  69871.42490157355
mae:  190.96440674857908
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

0.3.3 Tối ưu mô hình bằng việc tinh chỉnh (tunning) siêu tham số

[ ]:
```
#Liet ke các bo gia tri de tunning cho cac sieu tham so tuong ung cua mo hinh

param = {
         'max_depth':[2,3,4,5,6,7],
         'learning_rate':[0.15,0.1,0.05,0.01,0.005,0.001],
         'n_estimators':[100, 150,200,500,900,1200,1500],
         'n_estimators':[100,250,500,750,1000,1250,1500,1750],
         'min_samples_split':[2,4,6,8,10,20,40,60,100],
         'min_samples_leaf':[1,3,5,7,9],
         'max_features':[2,3,4,5,6,7],
         'subsample':[0.7,0.75,0.8,0.85,0.9,0.95,1]
         }
```

[ ]:
```
#thực hiện cross-validation để chọn bộ siêu tham số
from sklearn.model_selection import RandomizedSearchCV
random_cv = RandomizedSearchCV(estimator=clf,
                               param_distributions=param,
                               cv=5,n_iter=50,
                               scoring='neg_mean_absolute_error',n_jobs=4,
                               verbose=5,
                               return_train_score=True,
                               random_state=42)
```

[ ]:
```
random_cv.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits

[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done  10 tasks      | elapsed:  4.4min
[Parallel(n_jobs=4)]: Done  64 tasks      | elapsed: 15.8min
[Parallel(n_jobs=4)]: Done 154 tasks      | elapsed: 41.4min
[Parallel(n_jobs=4)]: Done 250 out of 250 | elapsed: 76.9min finished
```

[ ]:
```
RandomizedSearchCV(cv=5, error_score=nan,
                   estimator=GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0,
```

```
                                                    criterion='friedman_mse',
                                                    init=None,
                                                    learning_rate=0.1,
                                                    loss='ls', max_depth=5,
                                                    max_features=None,
                                                    max_leaf_nodes=None,
    min_impurity_decrease=0.0,
                                                    min_impurity_split=None,
                                                    min_samples_leaf=1,
                                                    min_samples_split=2,
    min_weight_fraction_leaf=0.0,
                                                    n_estimators=400,
                                                    n_…
                                                       0.005, 0.001],
                                            'max_depth': [2, 3, 4, 5, 6, 7],
                                            'max_features': [2, 3, 4, 5, 6, 7],
                                            'min_samples_leaf': [1, 3, 5, 7, 9],
                                            'min_samples_split': [2, 4, 6, 8, 10,
                                                                  20, 40, 60, 100],
                                            'n_estimators': [100, 250, 500, 750,
                                                             1000, 1250, 1500,
                                                             1750],
                                            'subsample': [0.7, 0.75, 0.8, 0.85, 0.9,
                                                          0.95, 1]},
                       pre_dispatch='2*n_jobs', random_state=42, refit=True,
                       return_train_score=True, scoring='neg_mean_absolute_error',
                       verbose=5)
```

[ ]:
```
#Lấy ra bộ tham số sau khi thực hiện CV (cross-validation)
random_cv.best_estimator_
```

[ ]:
```
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.15, loss='ls', max_depth=6,
                          max_features=5, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=4,
                          min_weight_fraction_leaf=0.0, n_estimators=1750,
                          n_iter_no_change=None, presort='deprecated',
                          random_state=None, subsample=0.85, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
```

[ ]:
```
#Xây dựng mô hình với bộ tham số vừa tìm được
clf2 = ensemble.GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0,␣
 ↪criterion='friedman_mse',
                          init=None, learning_rate=0.15, loss='ls', max_depth=6,
                          max_features=5, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
```

```
                           min_samples_leaf=1, min_samples_split=4,
                           min_weight_fraction_leaf=0.0, n_estimators=1750,
                           n_iter_no_change=None, presort='deprecated',
                           random_state=None, subsample=0.85, tol=0.0001,
                           validation_fraction=0.1, verbose=0, warm_start=False)
```

[ ]:
```
clf2.fit(x_train, y_train)
```

[ ]:
```
GradientBoostingRegressor(alpha=0.9, ccp_alpha=0.0, criterion='friedman_mse',
                          init=None, learning_rate=0.15, loss='ls', max_depth=6,
                          max_features=5, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=4,
                          min_weight_fraction_leaf=0.0, n_estimators=1750,
                          n_iter_no_change=None, presort='deprecated',
                          random_state=None, subsample=0.85, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
```

[ ]:
```
evaluate(clf2, x_train, x_test, y_train, y_test)
```

```
r2:  0.9461405284955543
mse:  59669.7469661674
mae:  153.89661794913926
----------------------------------------
r2:  0.9674391108010937
mse:  35847.10271607993
mae:  123.69941807247359
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

[ ]:
```
#Lưu mô hình

import pickle

pickle.dump(clf2, open('clf2.pkl', 'wb'))
```

[ ]:
```
model2 = pickle.load(open('clf2.pkl', 'rb'))
```